Polyteam Semantics

Hannula, Miika

Springer
2018

Hannula , M , Kontinen , J & Virtema , J 2018 , Polyteam Semantics . in S Artemov & A Nerode (eds) , Logical Foundations of Computer Science . vol. 2018 , Lecture Notes in Computer Science , vol. 10703 , Springer , pp. 190 210 , Logical Found Science 2018 , Deerfield Beach , United States , 08/01/2018 . https://doi.org/10.1007/978-3-319-72056-2_12

http://hdl.handle.net/10138/310094
https://doi.org/10.1007/978-3-319-72056-2_12

acceptedVersion

Polyteam Semantics [?]

Miika Hannula[1], Juha Kontinen[2], and Jonni Virtema[2;3]

[1] University of Auckland, New Zealand, m.hannula@auckland.ac.nz
[2] University of Helsinki, Finland, juha.kontinen@helsinki.fi
[3] Hasselt University, Belgium, jonni.virtema@uhasselt.be

Abstract. Team semantics is the mathematical framework of modern logics of dependence and independence in which formulae are interpreted by sets of assignments (teams) instead of single assignments as in first-order logic. In order to deepen the fruitful interplay between team semantics and database dependency theory, we define Polyteam Semantics in which formulae are evaluated over a family of teams. We begin by defining a novel polyteam variant of dependence atoms and give a finite axiomatisation for the associated implication problem. We also characterise the expressive power of poly-dependence logic by properties of polyteams that are downward closed and definable in existential second-order logic (ESO). The analogous result is shown to hold for poly-independence logic and all ESO definable properties.

1 Introduction

Team semantics is the mathematical framework of modern logics of dependence and independence. The origin of team semantics goes back to [15] but its development to its current form began with the publication of the monograph [22]. In team semantics formulae are interpreted by sets of assignments (teams) instead of single assignments as in first-order logic. The reason for this change is that statements such as the value of variable x depends on the value of y do not really make sense for single assignments. Team semantics has interesting connections with database theory and database dependencies [11,12,13,18]. In order to facilitate the exchange between team semantics and database theory, we introduce a generalisation of team semantics in which formulae are evaluated over a family of teams. We identify a natural notion of poly-dependence that generalises dependence atoms to polyteams and give a finite axiomatisation for its implication problem. We also define polyteam versions of independence, inclusion and exclusion atoms, and characterise the expressive power of poly-dependence and poly-independence logic.

A team X is a set of assignments with a common finite domain $x_1; \ldots; x_n$ of variables. Such a team can be viewed as a database table with $x_1; \ldots; x_n$ as its attributes. Dependence logic extends the language of first-order logic with atomic formulae $=(\bar{x}; y)$ called dependence atoms expressing that value of the variable y is functionally determined by the values of the variables in $\bar{x}$. On the other hand, independence atoms $\bar{y} \perp_{\bar{x}} \bar{z}$ [9] express that, for any fixed value of $\bar{x}$, knowing the value of $\bar{z}$ does not tell us anything new about the value of $\bar{y}$. By viewing a team as a database, the atoms $=(\bar{x}; y)$ and $\bar{y} \perp_{\bar{x}} \bar{z}$ correspond to the widely studied functional and embedded multivalued dependencies. Furthermore, inclusion atoms $\bar{x} \subseteq \bar{y}$ and exclusion atoms $\bar{x} | \bar{y}$ of [6] inherit their semantics from the corresponding database dependencies.

Independence, inclusion, and exclusion atoms have very interesting properties in the team semantics setting. For example, inclusion atoms give rise to a variant of dependence logic that corresponds to the complexity class PTIME over finite ordered structures [7] whereas all the other atoms above (and their combinations) give rise to logics that are equi-expressive with existential second-order logic and the complexity class NP. The complexity

[?] This research was supported by a Marsden grant from Government funding, administered by the Royal Society of New Zealand, and grants 292767 and 308712 of the Academy of Finland.

theoretic aspects of logics in team semantics have been studied extensively during the past few years (see [4] for a survey).

A multiset version of team semantics was recently de ned in [3]. Multiteam semantics is motivated by the fact that multisets are widely assumed in database theory and occur in applications. Multiteam semantics can be also used to model and study database, probabilistic, and approximate dependencies in a uni ed framework (see [3,23]).

The aim of this work is similar to that of [3], i.e., we want to extend the applicability of team semantics. In database theory dependencies are often expressed by so-called embedded dependencies. An embedded dependency is a sentence of rst-order logic with equality of the form

$$8x_1 \cdots 8x_n \quad (x_1; \ldots; x_n) \;!\; 9y_1 \cdots 9y_k \quad (x_1; \ldots; x_n; y_1; \ldots; y_k) \;;$$

where and are conjunctions of relational atoms $R(x_1; \ldots; x_n)$ and equalities $x = y$. In the literature embedded dependencies have been thoroughly classi ed stemming from real life applications. Examples of well-known subclasses include full, uni-relational, 1-head, tuple-generating, and equality-generating. For example, an embedded dependency is called tuple-generating if it is equality free (for further details see, e.g., [16, Section 3]). The uni-relational dependencies can be studied also in the context of team semantics as generalised dependencies [21]. However in many applications, especially in the are of data exchange and data integration, it is essential to be able to express dependencies between di erent relations.

In the context of data exchange (see e.g. [5]) the relational database is divided into a set of source relations S and a set of target relations T . Dependencies are used to describe what kind of properties should hold when data is transferred from the source schema to the target schema. In this setting a new taxonomy of embedded dependencies rises: An embedded dependency $8\bar{x} \; (\bar{x}) \;!\; 9\bar{y} \; (\bar{x}; \bar{y})$ is source-to-target if the relation symbols occurring in and are from S and T , respectively. The embedded dependency is target if the relation symbols occurring in it are from T . There is no direct way to study these classes of dependencies in the uni-relational setting of team semantics. In this paper we propose a general framework in which these inherently poly-relational dependencies can be studied.

In Section 2 we lay the foundations of polyteam semantics. The shift to polyteams is exempli ed in Section 2.2, by the de nition of poly-dependence atoms and an Armstrong type axiomatisation for the associated implication problem. In Section 3 polyteam semantics is extended from atoms to complex formulae. Section 4 studies the expressive power of the new logics over polyteams. The main technical results of the section characterises poly-independence (poly-dependence) logic as the maximal logic capable of de ning all (downward closed) properties of polyteams de nable in existential second-order logic.

2 From uni-dependencies to poly-dependencies

We start by de ning the familiar dependency notions from the team semantics literature. In Section 2:2 we introduce a novel poly-relational version of dependence atoms and establish a nite axiomatisation of its implication problem. We then continue to present poly-relational versions of inclusion, exclusion, and independence atoms, and a general notion of a poly-relational dependency atom. We conclude this section by relating the embedded dependencies studied in database theory to our new setting.

2.1 Dependencies in team semantics

Vocabularies are sets of relation symbols with prescribed arities. For each $R \in$, let $ar(R) \in \mathbb{Z}_+$ denote the arity of R. A -structure is a tuple $A = \; A; (R_i^A)_{R_i \in}$, where A is

a set and each R_i^A is an $ar(R_i)$-ary relation on A (i.e., $R_i^A \subseteq A^{ar(R_i)}$). We use A, B, etc. to denote τ-structures and A, B, etc. to denote the corresponding domains.

Let D be a finite set of first-order variables and A be a nonempty set. A function $s: D \to A$ is called an assignment. For a variable x and $a \in A$, the assignment $s(a/x): D \cup \{x\} \to A$ is obtained from s as follows:

$$s(a/x)(y) := \begin{cases} a & \text{if } y = x; \\ s(y) & \text{otherwise:} \end{cases}$$

For an assignment s and a tuple of variables $\vec{x} = (x_1, \ldots, x_n)$, we write $s(\vec{x})$ to denote the sequence $s(x_1), \ldots, s(x_n)$. A team is a set of assignments with a common domain D and codomain A. Let A be a τ-structure and X a team with codomain A, then we say that X is a team of A.

The following dependency atoms were introduced in [22,6,9].

Definition 1 (Dependency atoms). Let A be a model and X a team with codomain A. If $\vec{x}, \vec{y}$ are variable sequences, then $=(\vec{x}, \vec{y})$ is a dependence atom with the truth condition:

$$A \models_X =(\vec{x}, \vec{y}) \text{ if for all } s, s' \in X \text{ s.t. } s(\vec{x}) = s'(\vec{x}), \text{ it holds that } s(\vec{y}) = s'(\vec{y}).$$

If $\vec{x}, \vec{y}$ are variable sequences of the same length, then $\vec{x} \subseteq \vec{y}$ is an inclusion atom and $\vec{x} \mid \vec{y}$ an exclusion atom with satisfaction defined as follows:

$$A \models_X \vec{x} \subseteq \vec{y} \text{ if for all } s \in X \text{ there exists } s' \in X \text{ such that } s(\vec{x}) = s'(\vec{y}).$$
$$A \models_X \vec{x} \mid \vec{y} \text{ if for all } s, s' \in X : s(\vec{x}) \neq s'(\vec{y}).$$

If $\vec{x}, \vec{y}, \vec{z}$ are variable sequences, then $\vec{y} \perp_{\vec{x}} \vec{z}$ is a conditional independence atom with satisfaction defined by

$$A \models_X \vec{y} \perp_{\vec{x}} \vec{z} \text{ if for all } s, s' \in X \text{ such that } s(\vec{x}) = s'(\vec{x}) \text{ there exists } s'' \in X$$
$$\text{such that } s''(\vec{x}) = s(\vec{x}),\ s''(\vec{y}) = s(\vec{y}), \text{ and } s''(\vec{z}) = s'(\vec{z}).$$

Note that in the previous definitions it is allowed that some or all of the vectors of variables have length 0. For example, $A \models_X =(\vec{x})$ holds iff $\forall s \in X : s(\vec{x}) = \vec{c}$ holds for some fixed tuple $\vec{c}$, and $A \models_X \vec{y} \perp_{\vec{x}} \vec{z}$ holds always if either of the vectors $\vec{y}$ or $\vec{z}$ is of length 0.

All the aforementioned dependency atoms have corresponding variants in relational databases. One effect of this relationship is that the axiomatic properties of these dependency atoms trace back to well-known results in database theory. Armstrong's axioms for functional dependencies constitute a finite axiomatisation for dependence atoms [1,9], and inclusion atoms can be finitely axiomatised using the axiomatisation for inclusion dependencies [2]. Furthermore, by the undecidability of the (finite and unrestricted) implication problem for embedded multivalued dependencies the same limitation applies to conditional independence atoms as well [14]. Restricting attention to the so-called pure independence atoms, i.e., atoms of the form $\vec{x} \perp_{\vec{z}} \vec{y}$, a finite axiomatisation is obtained by relating to marginal independence in statistics [8,18].

2.2 The notion of poly-dependence

For each $i \in \mathbb{N}$, let Var(i) denote a distinct countable set of first-order variable symbols. We say that these variables are of sort i. Relating to databases, sorts correspond to table names. Usually we set $Var(i) = \{x_j^i \mid j \in \mathbb{N}\}$. We write x^i, y^i, x_j^i to denote variables form

Var(i), and $\vec{x}^i$ to denote tuples of variables from Var(i). Sometimes we drop the index i and write simply x and $\vec{x}$ instead of x^i and $\vec{x}^i$, respectively. Note that $\vec{x}$ is always a tuple of variables of a single sort. In order to simplify notation, we sometimes write $\vec{x}^i$ and $\vec{x}^j$ to denote arbitrary tuples of variables of sort i and j, respectively. We emphasise that $\vec{x}^i$ and $\vec{x}^j$ might be of di erent length and may consist of distinct variables. Let A be a -model and let D_i Var(i) for all i 2 N. A tuple $\overline{X} = (X_i)_{i2N}$ is a polyteam of A with domain $\overline{D} = (D_i)_{i2N}$, if X_i is a team with domain D_i and co-domain A for each i 2 N. We identify $\overline{X}$ with $(X_0; : : : ; X_n)$ if X_i is the singleton team consisting with the empty assignment for all i greater than n. Let $\overline{X} = (X_i)_{i2N}$ and $\overline{Y} = (Y_i)_{i2N}$ be two polyteams. We say that $\overline{X}$ is a subteam of $\overline{Y}$ if X_i Y_i for all i 2 N. By the union (resp. intersection) of $\overline{X}$ and $\overline{Y}$ we denote the polyteam $(X_i [Y_i)_{i2N}$ (resp. $(X_i \setminus Y_i)_{i2N}$). By a slight abuse of notation we write $\overline{X} [\overline{Y}$ (resp. $\overline{X} \setminus \overline{Y}$) for the union (resp. intersection) of $\overline{X}$ and $\overline{Y}$, and $\overline{X}$ $\overline{Y}$ to denote that $\overline{X}$ is a subteam of $\overline{Y}$. For a tuple $\overline{V} = (V_i)_{i2N}$ where V_i Var(i), the restriction of $\overline{X}$ to $\overline{V}$, written $\overline{X}$ $\overline{V}$, is de ned as $(X_i$ $V_i)_{i2N}$ where X_i V_i denotes the restriction of X_i to V_i.

Next we generalise dependence atoms to the polyteam setting. In contrast to the standard dependence atoms, poly-dependence atoms declare functional dependence of variables over two teams.

Poly-dependence. Let $\vec{x}^i\vec{y}^i$ and $\vec{u}^j\vec{v}^j$ be sequences of variables such that $\vec{x}^i$ and $\vec{u}^j$, and $\vec{y}^i$ and $\vec{u}^j$ have the same length, respectively. Then $= \vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{v}^j$ is a poly-dependence atom whose satisfaction relation $\models_{\overline{X}}$ is de ned as follows:

$$A \models_{\overline{X}} = \vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{v}^j \quad , \quad 8s \in X_i 8s' \in X_j : s(\vec{x}^i) = s'(\vec{u}^j) \text{ implies } s(\vec{y}^i) = s'(\vec{v}^j):$$

Note that the atom $= (\vec{x}; \vec{y} = \vec{x}; \vec{y})$ corresponds to the dependence atom $= (\vec{x}; \vec{y})$. For empty tuples $\vec{x}^i$ and $\vec{u}^j$ the poly-dependence atom reduces to a "poly-constancy atom" $= \vec{y}^i = \vec{v}^j$. We will later show (Remark 13) that poly-dependence atoms of the form $= \vec{x}^i; \vec{y}^i = \vec{u}^i; \vec{v}^i$ can be expressed with formulae using only ordinary dependence atoms. Thus poly-dependence atoms of this form are considered as primitive notions only when $\vec{x}^i\vec{y}^i = \vec{u}^i\vec{v}^i$; otherwise $= \vec{x}^i; \vec{y}^i = \vec{u}^i; \vec{v}^i$ is considered as a shorthand for the equivalent formula obtained from Remark 13.

The ability to reason about database dependencies can be employed to facilitate many critical data management tasks such as schema design, query optimisation, and integrity maintenance. Keys, inclusion dependencies, and functional dependencies in particular have a crucial role in all of these processes. A traditional way to approach the interaction between dependencies has been the utilisation of proof systems similar to natural deduction systems in logic. The most signi cant of all these systems is the Armstrong's axiomatisation for functional dependencies. This inference system consists of only three rules which we depict below using the standard notation for functional dependencies, i.e., $X \to Y$ denotes that an attribute set X functionally determines another attribute set Y.

De nition 2 (Armstrong's axiomatisation [1]).

- Re exivity: If $Y \subseteq X$, then $X \to Y$
- Augmentation: if $X \to Y$, then $XZ \to YZ$
- Transitivity: if $X \to Y$ and $Y \to Z$, then $X \to Z$

Our rst development is the generalisation of Armstrong's axiomatisation to the poly-dependence setting. To this end, we assemble the three rules of Armstrong and introduce three auxiliary rules: Union, Symmetry, and Weak Transitivity. Contrary to the Armstrong's proof system, here Union is not reducible to Transitivity and Augmentation because we operate with sequences instead of sets of variables or attributes. Symmetry in turn is imposed by

the sequential notation employed by the poly-dependence atom. Weak Transitivity exhibits transitivity of equalities on the right-hand side of a poly-dependence atom, a phenomenon that arises only in the polyteam setting.

De nition 3 (Axiomatisation for poly-dependence atoms).

- { Re exivity: = $\vec{x}^i; \mathrm{pr}_k(\vec{x}^i) = \vec{y}^j; \mathrm{pr}_k(\vec{y}^j)$, where $k = 1; \ldots; |\vec{x}^i|$ and pr_k takes the kth projection of a sequence.
- { Augmentation: if = $\vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{v}^j$, then = $\vec{x}^i\vec{z}^i; \vec{y}^i\vec{z}^i = \vec{u}^j\vec{w}^j; \vec{v}^j\vec{w}^j$
- { Transitivity: if = $\vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{v}^j$ and = $\vec{y}^i; \vec{z}^i = \vec{v}^j; \vec{w}^j$, then = $\vec{x}^i; \vec{z}^i = \vec{u}^j; \vec{w}^j$
- { Union: if = $\vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{v}^j$ and = $\vec{x}^i; \vec{z}^i = \vec{u}^j; \vec{w}^j$ then = $\vec{x}^i; \vec{y}^i\vec{z}^i = \vec{u}^j; \vec{v}^j\vec{w}^j$
- { Symmetry: if = $\vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{v}^j$, then = $\vec{u}^j; \vec{v}^j = \vec{x}^i; \vec{y}^i$
- { Weak Transitivity: if = $\vec{x}^i; \vec{y}^i\vec{z}^i = \vec{u}^j; \vec{v}^j\vec{v}^j\vec{w}^j$, then = $\vec{x}^i; \vec{y}^i = \vec{u}^j; \vec{w}^j$

This proof system forms a complete characterisation of logical implication for poly-dependence atoms. We use $\models$ to refer to logical implication, i.e., we write $\models$ if $A \models_X$ implies $A \models_X$ for all models A and polyteams X. Given an axiomatisation R, that is, a set of axioms and inference rules, we write $\vdash_R$ if R yields a proof of from . Given a class of dependency atoms C, we then say that R is sound (complete, resp.) for C if for all nite sets of dependency atoms $\cup \{\}$ from C, $\vdash_R$ implies (is implied by, resp.) $\models$.

Theorem 4. The axiomatisation of Def. 3 is sound and complete for poly-dependence atoms.

Proof. The proof of soundness is straightforward and omitted. We show that the axiomatisation is complete, i.e., that $\models$ implies $\vdash$ for a set $\cup \{\}$ of poly-dependence atoms. Assume is $\vec{x}^i; \vec{y}^i = \vec{x}^j; \vec{y}^j$. First we consider the case where $i = j$ in which case is a standard dependence atom. Let be the subset of consisting of all standard dependence atoms over Var(i). Since all teams satisfying can be extended to a polyteam satisfying by introducing new empty teams, we have that $\models$ in the team semantics setting. Since dependence atoms $= (\vec{x}; \vec{y})$ in team semantics correspond to functional dependencies $\{x \in \vec{x}^i\} \to \{y \in \vec{y}^i\}$ in relational databases (see e.g. [9]), Armstrong's complete axiomatisation from De nition 2 yields a deduction of $_0$ from $_0$ where $_0$ and $\{_0\}$ are obtained from and by replacing dependence atoms with their corresponding functional dependencies. Since dependence atoms are provably order-independent (i.e. one derives $= (\vec{x}_0; \vec{x}_1)$ from $= (\vec{y}_0; \vec{y}_1)$ by Re exivity, Union, and Transitivity if $\vec{x}_i$ and $\vec{y}_i$ list the same variables), the deduction in Armstrong's system can be simulated with the rules in De nition 3. This proves the case $i = j$.

Let us then consider the case $i \neq j$. We will show that $\not\vdash$ implies $\not\models$. Assume $\not\vdash$. De ne rst a binary relation on Var(i) [Var(j) such that $a^i \; a^j$ if $' = \vec{x}^i; a^i = \vec{x}^j; a^j$, $a^j \; a^i$ if $' = \vec{x}^j; a^j = \vec{x}^i; a^i$, and $a^i \; b^i$ ($a^j \; b^j$, resp.) if $a^i = b^i$ or $' = \vec{x}^i; a^ib^i = \vec{x}^j; a^ja^j$ for some a^j ($a^j = b^j$ or $' = \vec{x}^j; a^jb^j = \vec{x}^i; a^ia^i$ for some a^i, resp.). We show that is an equivalence relation.

- { Re exivity: Holds by de nition.
- { Symmetry: First note that $a^i \; a^j$ and $a^j \; a^i$ are derivably equivalent by the symmetry rule. Assume that $a^i \; b^i$ in which case $= \vec{x}^i; a^ib^i = \vec{x}^j; a^ja^j$ is derivable for some a^j. Then derive rst $= (a^ib^i; b^i = a^ja^j; a^j)$ and $= (a^ib^i; a^i = a^ja^j; a^j)$ by using the re exivity rule, and then $= \vec{x}^i; b^i = \vec{x}^j; a^j$ and $= \vec{x}^i; a^i = \vec{x}^j; a^j$ by using the transitivity rule. Finally derive $= \vec{x}^i; b^ia^i = \vec{x}^j; a^ja^j$ by using the union rule.

{ Transitivity: Assume first that $a^i \; b^i \; c^i$, where $a^i; b^i; c^i$ and are pairwise distinct. Then $= \bar{x}^i; a^i b^i = \bar{x}^j; a^j a^j$ and $= \bar{x}^i; b^i c^i = \bar{x}^j; b^j b^j$ are derivable for some a^j and b^j. Then analogously to the previous case assemble $= \bar{x}^i; a^i b^i b^i = \bar{x}^j; a^j a^j b^j$ which admits $= \bar{x}^i; a^i = \bar{x}^j; b^j$ by weak transitivity, and detach $= \bar{x}^i; c^i = \bar{x}^j; b^j$ from $= \bar{x}^i; b^i c^i = \bar{x}^j; b^j b^j$. By the union rule we then obtain $= \bar{x}^i; a^i c^i = \bar{x}^j; b^j b^j$ and thus that $a^i \; c^i$. Since all the other cases are analogous, we observe that is transitive.

Let s be a function that maps each $x \in Var(i) \cup Var(j)$ that appears in $\cup \{ \}$ to the equivalence class $x/$. We define $\overline{X} = (X_i; X_j)$ where $X_k = \{s \; Var(k)\}$ for $k = i; j$. First notice that $\overline{X} \neq$ for, by union, it cannot be the case that $pr_k(\bar{y}^i) \; pr_k(\bar{y}^j)$ for all $k = 1; \ldots; |\bar{y}^i|$. It suffices to show that $\overline{X}$ satisfies each $= (\bar{u}^m; \bar{v}^m = \bar{u}^n; \bar{v}^n)$ in . If $m = n$ or $\{m; n\} \neq \{i; j\}$, the atom is trivially satisfied. Hence, and by symmetry, we may assume that the atom is of the form $= \bar{u}^i; \bar{v}^i = \bar{u}^j; \bar{v}^j$. Assume that $s(\bar{u}^i) = s(\bar{u}^j)$, that is, $pr_k(\bar{u}^i) \; pr_k(\bar{u}^j)$ for all $k = 1; \ldots; |\bar{u}^i|$. We obtain by the union rule that $= \bar{x}^i; \bar{u}^i = \bar{x}^j; \bar{u}^j$ is derivable, and hence by the transitivity rule that $= \bar{x}^i; \bar{v}^i = \bar{x}^j; \bar{v}^j$ is also derivable. Therefore, by using the reflexivity and transitivity rules we conclude that $s(\bar{v}^i) = s(\bar{v}^j)$. □

2.3 A general notion of a poly-dependency

Next we consider suitable polyteam generalisations for the dependencies discussed in Section 2.1 and also define a general notion of poly-dependency. This generalisation is immediate for inclusion atoms which are inherently multi-relational; relational database management systems maintain referential integrity by enforcing inclusion dependencies specifically between two distinct tables. With poly-inclusion atoms these multi-relational features can now be captured.

Poly-inclusion. Let $\bar{x}^i$ and $\bar{y}^j$ be sequences of variables of the same length. Then $\bar{x}^i \; \bar{y}^j$ is a poly-inclusion atom whose satisfaction relation $\models_{\overline{X}}$ is defined as follows:

$$A \models_{\overline{X}} \bar{x}^i \; \bar{y}^j \quad , \quad \forall s \in X_i \exists s' \in X_j : s(\bar{x}^i) = s'(\bar{y}^j).$$

If $i = j$, then the atom is the standard inclusion atom.

Poly-exclusion. Let $\bar{x}^i$ and $\bar{y}^j$ be sequences of variables of the same length. Then $\bar{x}^i \mid \bar{y}^j$ is a poly-exclusion atom whose satisfaction relation $\models_{\overline{X}}$ is defined as follows:

$$A \models_{\overline{X}} \bar{x}^i \mid \bar{y}^j \quad , \quad \forall s \in X_i; s' \in X_j : s(\bar{x}^i) \neq s'(\bar{y}^j).$$

If $i = j$, then the atom is the standard exclusion atom.

Poly-independence Let $\bar{x}^i$, $\bar{y}^i$, $\bar{a}^j$, $\bar{b}^j$, $\bar{u}^k$, $\bar{v}^k$, and $\bar{w}^k$ be tuples of variables such that $|\bar{x}^i| = |\bar{a}^j| = |\bar{u}^k|$, $|\bar{y}^i| = |\bar{v}^k|$, $|\bar{b}^j| = |\bar{w}^k|$. Then $\bar{y}^i / \bar{v}^k \; ?_{\bar{x}^i; \bar{a}^j = \bar{u}^k} \; \bar{b}^j = \bar{w}^k$ is a poly-independence atom whose satisfaction relation $\models_{\overline{X}}$ is defined as follows:

$$A \models_{\overline{X}} \bar{y}^i / \bar{v}^k \; ?_{\bar{x}^i; \bar{a}^j = \bar{u}^k} \; \bar{b}^j = \bar{w}^k \quad , \quad \forall s \in X_i; s' \in X_j : s(\bar{x}^i) = s'(\bar{a}^j) \text{ implies}$$
$$\exists s'' \in X_k : s''(\bar{u}^k \bar{v}^k) = s(\bar{x}^i \bar{y}^i) \text{ and } s''(\bar{w}^k) = s'(\bar{b}^j).$$

The atom $\bar{y} / \bar{y} \; ?_{\bar{x}; \bar{x} = \bar{x}} \; \bar{z} = \bar{z}$, where all variables are of the same sort, corresponds to the standard independence atom $\bar{y} \; ?_{\bar{x}} \; \bar{z}$. Furthermore, a pure poly-independence atom is an atom of the form $\bar{y}^i / \bar{v}^k \; ?_{;;;=;} \; \bar{b}^j = \bar{w}^k$, written using a shorthand $\bar{y}^i / \bar{v}^k \; ? \; \bar{b}^j = \bar{w}^k$.

Poly-independence atoms are closely related to equi-join operators of relational databases as the next example exemplifies.

Example 5. A relational database schema

$$P(\text{rojects}) = \{\text{project,team } g; \quad T(\text{eams}) = \{\text{team,employee}g;$$
$$E(\text{mployees}) = \{\text{employee,team,project } g;$$

stores information about distribution of employees for teams and projects in a workplace. The poly-independence atom

$$P[\text{project }]=E[\text{project }] \; ?_{P[\text{team}];T[\text{team}]=E[\text{team}]} \; T[\text{employee}]=E[\text{employee}] \qquad (1)$$

expresses that the relation Employees includes as a subrelation the natural join of Projects and Teams. If furthermore $E[\text{project,team }] \quad P[\text{project,team }]$ and $E[\text{team,employee}] \quad T[\text{team,employee}]$ hold, then Employees is exactly this natural join.

In addition to the poly-atoms described above we de ne a notion of a generalised poly-atom, similarly to the notion of generalised atom of [21].

Generalised poly-atoms. Let $(j_1; : : : ; j_n)$ be a sequence of positive integers. A generalised quanti er of type $(j_1; : : : ; j_n)$ is a collection Q of relational structures $(A; R_1; : : : ; R_n)$ (where each R_i is j_i-ary) that is closed under isomorphisms. Then, for any sequence $(\bar{x}_1; : : : ; \bar{x}_n)$ where $\bar{x}_i$ is a length j_i tuple of variables from some $\mathrm{Var}(l_i)$, $A_Q(\bar{x}_1; : : : ; \bar{x}_n)$ is a generalised poly-atom of type $(j_1; : : : ; j_n)$. For a model A and polyteam $\overline{X}$ where $\bar{x}_i \quad \mathrm{Dom}(X_{l_i})$, the satisfaction relation with respect to A_Q is de ned as follows:

$$A \models_{\overline{X}} A_Q(\bar{x}_1; : : : ; \bar{x}_n) \quad , \quad \mathrm{Dom}(A); R_1 := \mathrm{rel}(X_{l_1}; (\bar{x}_1)) : : : ; R_n := \mathrm{rel}(X_{l_n}; (\bar{x}_n)) \quad 2 \; Q:$$

Note that by $\mathrm{rel}(X; (\bar{x}))$ for $\bar{x} = (x_1; : : : ; x_m)$ we denote the relation $f(s(x_1); : : : ; s(x_m)) \; j \; s \; 2 \; Xg$. A poly-atom $A_Q(\bar{x}_1; : : : ; \bar{x}_n)$ is a uni-atom if the variables sequences $\bar{x}_1; : : : ; \bar{x}_n$ are of a single sort. Uni-atoms correspond exactly to generalised atoms of [21]. We say that the atom $A_Q(\bar{x}_1; : : : ; \bar{x}_n)$ is de nable in a logic L if the class Q is de nable in L. For instance, we notice that a poly-inclusion atom $(x^1; y^1) \quad (u^2; v^2)$ is a rst-order de nable generalised poly-atom of type (2; 2).

2.4 Database dependencies as poly-atoms

Embedded dependencies in a multi-relational context can now be studied with the help of generalised poly-atoms and polyteam semantics. Conversely, strong results obtained in the study of database dependencies can be transferred and generalised for stronger results in the polyteam setting. In particular, each embedded dependency can be seen as a de ning formula for a generalised poly-atom, and hence the classi cation of embedded dependencies naturally yield a corresponding classi cation of generalised poly-atoms. For example, the class

$$C := \{A_Q(\bar{x}_1; : : : ; \bar{x}_n) \; j \; Q \text{ is de nable by an FO}(R_1; : : : ; R_n)\text{-sentence in the class of equality-generating dependencies}g$$

is the class of equality-generating poly-atoms. The de ning formula of the generalised atom of type (2,2) that captures the poly-dependence atom of type $= (x^i; y^i / u^j; v^j)$ is

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 \; (R_1(x_1; x_2) \wedge R_2(y_1; y_2) \wedge x_1 = y_1) \; ! \; x_2 = y_2 \; :$$

Thus poly-dependence atoms are included in the class of equality-generating poly-atoms.

In order to study data exchange in the polyteam setting, we rst need to de ne the notions of source-to-target and target poly-atoms. This classi cation of poly-atoms requires some more care as it is not enough to consider the de ning formulae of the corresponding atoms, but also the variables that the atom is instantiated with. We will return to this topic brie y after we have given semantics for logics that work on polyteams.

3 Polyteam semantics for complex formulae

We next delineate a version of team semantics suitable for the polyteam context. We note here that it is not a priori clear what sort of modi cations for connectives and quanti ers one should entertain when shifting from teams to the polyteam setting.

3.1 Syntax and semantics

De nition 6. Let be a set of relation symbols. The syntax of poly rst-order logic PFO( ) is given by the following grammar rules:

$$::= x = y \mid x \neq y \mid R(\vec{x}) \mid :R(\vec{x}) \mid (\ \wedge \ ) \mid (\ \vee \ ) \mid (\ \vee^{j} \ ) \mid \exists x \mid \forall x \ ;$$

where $R \in$ is a k-ary relation symbol, $j \in \mathbb{N}$, $\vec{x} \in \mathrm{Var}(i)^k$ and $x, y \in \mathrm{Var}(i)$ for some $i, k \in \mathbb{N}$.

We say that $\vee$ is a global disjunction whereas $\vee^i$ is a local disjunction. Note that in the de nition the scope of negation is restricted to atomic formulae. Note also that the restriction of PFO( ) to formulae without the connective $\vee^j$ and using only variables of a single xed sort is FO( ).

For the de nition of the polyteam semantics of PFO, recall the de nitions of teams and polyteams from Sections 2.1 and 2.2, respectively. Let X be a team, A a nite set, and $F : X \to \mathcal{P}(A) \setminus \{\emptyset\}$ a function. We denote by $X[A/x]$ the modi ed team $\{s(a/x) \mid s \in X, a \in A\}$, and by $X[F/x]$ the team $\{s(a/x) \mid s \in X, a \in F(s)\}$. Again note that if restricted to the above fragment of PFO( ) the polyteam semantics below coincides with traditional team semantics, see e.g. [4] for a de nition. Thus for FO( ) formulae we may write $A \models_{X_i}$ instead of $A \models_{(X_i)}$.

De nition 7 (Lax polyteam semantics). Let A be a -structure and $\overline{X}$ a polyteam of A. The satisfaction relation $\models_{\overline{X}}$ for poly rst-order logic is de ned as follows:

$A \models_{\overline{X}} x = y$, if $x, y \in \mathrm{Var}(i)$ then $\forall s \in X_i : s(x) = s(y)$
$A \models_{\overline{X}} x \neq y$, if $x, y \in \mathrm{Var}(i)$ then $\forall s \in X_i : s(x) \neq s(y)$
$A \models_{\overline{X}} R(\vec{x})$, if $\vec{x} \in \mathrm{Var}(i)^k$ then $\forall s \in X_i : s(\vec{x}) \in R^A$
$A \models_{\overline{X}} :R(\vec{x})$, if $\vec{x} \in \mathrm{Var}(i)^k$ then $\forall s \in X_i : s(\vec{x}) \notin R^A$
$A \models_{\overline{X}} (\ \wedge \ )$, $A \models_{\overline{X}}$ and $A \models_{\overline{X}}$
$A \models_{\overline{X}} (\ \vee \ )$, $A \models_{\overline{Y}}$ and $A \models_{\overline{Z}}$ for some $\overline{Y}, \overline{Z} \subseteq \overline{X}$ s.t. $\overline{Y} \cup \overline{Z} = \overline{X}$
$A \models_{\overline{X}} (\ \vee^{j} \ )$, $A \models_{\overline{X}[Y_j/X_j]}$ and $A \models_{\overline{X}[Z_j/X_j]}$ for some $Y_j, Z_j \subseteq X_j$ s.t. $Y_j \cup Z_j = X_j$
$A \models_{\overline{X}} \forall x$, $A \models_{\overline{X}[X_i[A/x]/X_i]}$, when $x \in \mathrm{Var}(i)$
$A \models_{\overline{X}} \exists x$, $A \models_{\overline{X}[X_i[F/x]/X_i]}$ holds for some $F : X_i \to \mathcal{P}(A) \setminus \{\emptyset\}$, when $x \in \mathrm{Var}(i)$.

The truth of a sentence (i.e., a formula with no free variables) in a model A is de ned as: $A \models$ if $A \models_{(\{\emptyset\})}$; where $(\{\emptyset\})$ denotes the polyteam consisting only singleton teams of the empty assignment. We write $\mathrm{Fr}( )$ for the set of free variables in , and $\mathrm{Fr}_i( )$ for $\mathrm{Fr}( ) \setminus \mathrm{Var}(i)$.

Polyteam semantics is a conservative extension of team semantics in the same fashion as teams semantics is a conservative extension of Tarski semantics [22].

Proposition 8. Let $\in \mathrm{FO}( )$ whose variables are all of sort $i \in \mathbb{N}$. Let A be a -structure and $\overline{X}$ a polyteam of A. Then $A \models_{\overline{X}}$, $A \models_{X_i}$, $\forall s \in X_i : A \models_s$; where $\models_s$ denotes the ordinary satisfaction relation of rst-order logic.

Example 9. A relational database schema

$$\text{Patient} = \{\text{patient\_id}, \text{patient\_name}\};$$
$$\text{Case} = \{\text{case\_id}, \text{patient\_id}, \text{diagnosis\_id}, \text{confirmation}\};$$
$$\text{Test} = \{\text{diagnosis\_id}, \text{test\_id}\}; \quad \text{Results} = \{\text{patient\_id}, \text{test\_id}, \text{result}\}$$

stores information about patient cases and their related laboratory tests. In order to maintain consistency of the stored data, database management systems support the use of integrity constraints that are based on functional and inclusion dependencies. For instance, on relation schema Patient the key patient_id (i.e. the dependence atom $=(\text{patient\_id}, \text{patient\_name})$) ensures that no patient id can refer to two different patient names. On Case the foreign key patient_id referring to patient_id on Patient (i.e. the inclusion atom $\text{Case}[\text{patient\_id}] \subseteq \text{Patient}[\text{patient\_id}]$) enforces that patient ids on Case refer to real patients. The introduction of poly-dependence logics opens up possibilities for more expressive data constraints. The poly-inclusion formula

$$\psi_0 = \text{confirmation} \not\subseteq \text{positive}_{\text{Case}} \exists x_1 x_2 \; x_1 \not\subseteq x_2 \wedge \bigwedge_{i=1,2} (\text{Case}[\text{diagnosis\_id}; x_i] \subseteq \text{Test}[\text{diagnosis\_id}, \text{test\_id}] \wedge \text{Case}[\text{patient\_id}; x_i; \text{positive}] \subseteq \text{Results}[\text{patient\_id}, \text{test\_id}, \text{result}])$$

ensures that a diagnosis may be confirmed only if it has been affirmed by two different appropriate tests. The poly-exclusion formula

$$\psi_1 = \text{confirmation} \not\subseteq \text{negative}_{\text{Case}} \; \forall x \; \text{Case}[\text{diagnosis\_id}; x] \mid \text{Test}[\text{diagnosis\_id}, \text{test\_id}]_{\text{Case}} \; \text{Case}[\text{patient\_id}; x_i; \text{positive}] \mid \text{Results}[\text{patient\_id}, \text{test\_id}, \text{result}]$$

makes sure that a diagnosis may obtain a negative confirmation only if it has no positive indication by any suitable test. Note that both formulae employ local disjunction and quantified variables that refer to Case. Interestingly, the illustrated expressive gain is still computationally feasible as both ψ_0 and ψ_1 can be enforced in polynomial time. For ψ_0 note that the data complexity of inclusion logic is in PTIME [7]; for ψ_1 observe that satisfaction of a formula of the form $\vec{x}^1 \mid \vec{y}^2 \; \__1 \; \vec{x}^1 \mid \vec{z}^3$ can be decided in PTIME as well.

Poly-dependence logics. Poly-dependence, poly-independence, poly-inclusion, and poly-exclusion logics (PFO(pdep), PFO(pind), PFO(pinc), and PFO(pexc), resp.) are obtained by extending PFO with poly-dependence, poly-independence, poly-inclusion, and poly-exclusion atoms, respectively. In general, given a set of atoms C we denote by PFO(C) the logic obtained by extending PFO with the atoms of C. We also consider poly-atoms in the team semantics setting; by FO(C) we denote the extension of first-order logic by the poly-atoms in C. Similarly, it is also possible to consider atoms of Section 2.1 in the polyteam setting by requiring that the variables used with each atom are of a single sort.

3.2 Basic properties

We say that a formula α is local in polyteam semantics if for all $\overline{V} = (V_i)_{i \in N}$ where $Fr_i(\alpha) \subseteq V_i$ for $i \in N$, and all models $\mathcal{A}$ and polyteams $\overline{X}$, we have

$$\mathcal{A} \models_{\overline{X}} \alpha \quad , \quad \mathcal{A} \models_{\overline{X}\,\overline{V}} \alpha \; .$$

In other words, the truth value of a local formula depends only on its free variables. Furthermore, a logic L is called local if all its formulae are local.

Proposition 10 (Locality). For any set C of generalised poly-atoms PFO(C) is local.

Furthermore, the downward closure of dependence logic as well as the union closure of inclusion logic generalise to polyteams.

Proposition 11 (Downward Closure and Union Closure). Let ϕ be a formula of PFO(pdep), ψ a formula of PFO(pinc), $\mathcal{A}$ a model, and $\overline{X}, \overline{Y}$ two polyteams. Then $\mathcal{A} \models_{\overline{X}} \phi$ and $\overline{Y} \subseteq \overline{X}$ implies that $\mathcal{A} \models_{\overline{Y}} \phi$, and $\mathcal{A} \models_{\overline{X}} \psi$ and $\mathcal{A} \models_{\overline{Y}} \psi$ implies that $\mathcal{A} \models_{\overline{X} \cup \overline{Y}} \psi$.

The following proposition shows that the substitution of independence (dependence) atoms for any (downwards closed) class of atoms definable in existential second-order logic (ESO) results in no expressive gain.

Proposition 12. Let C (D, resp.) be the class of all (all downward closed, resp.) ESO-definable poly-atoms. The following equivalences of logics hold: FO(C) $\equiv$ FO(ind), FO(D) $\equiv$ FO(dep), and FO(pinc) $\equiv$ FO(inc).

Proof. The claim FO(pinc) $\equiv$ FO(inc) follows directly from the observation that in the team semantics setting poly-inclusion atoms are exactly inclusion atoms. Note that FO(ind) (FO(dep), resp.) captures all (all downward closed, resp.) ESO-definable properties of teams (see Theorem 18). It is easy to show (cf. [17, Theorem 6]) that every property of teams definable in FO(C) (FO(D), resp.) is ESO-definable (ESO-definable and downward closed, resp.). Thus since ind $\in C$ and dep $\in D$, we obtain that FO(C) $\equiv$ FO(ind) and FO(D) $\equiv$ FO(dep). □

Remark 13. In particular it follows from the previous proposition that, in the polyteam setting, each occurrence of any (any downward closed, resp.) ESO-definable poly-atom that takes variables of a single sort as parameters may be equivalently expressed by a formula of PFO(ind) (PFO(dep), resp.) that only uses variables of the same single sort.

We end this section by considering the relationship of global and local disjunctions. In particular, we observe that by the introduction of local disjunction its global variant becomes redundant. To facilitate our construction we here allow the use of $\mathbin{|}_I$, where I is a set on indices, with obvious semantics. We then show that $\mathbin{|}$ can be replaced by $\mathbin{|}_I$ and $\mathbin{|}_I$ by $\mathbin{|}_i$.

Proposition 14. For every formula ϕ of PFO there exists an equivalent formula ϕ' of PFO that only uses disjunctions of type $\mathbin{|}_i$.

Proof. Let ϕ be a formula of PFO and let I list the sorts of all the variables that occur in ϕ. Let ϕ' denote the formula obtained from ϕ by substituting all occurrences of $\mathbin{|}$ by $\mathbin{|}_I$. It is a direct consequence of the locality property that ϕ and ϕ' are equivalent.

We will next show how to eliminate disjunctions of type $\mathbin{|}_I$ from ϕ'. Let $\psi_0 \mathbin{|}_I \psi_1$ be a formula of PFO and let $I = \{i_1, \ldots, i_n\}$. Define

$$:= \forall z_0^{i_1} \forall z_1^{i_1} \ldots \forall z_0^{i_n} \forall z_1^{i_n} (\psi_0 \wedge \psi_1);$$

where $z_0^{i_1}$, $z_1^{i_1}$, ..., $z_0^{i_n}$, $z_1^{i_n}$ are fresh and distinct variables, and

$$\psi_0 := (z_0^{i_1} = z_1^{i_1} \mathbin{|}_{i_1} (z_0^{i_1} \neq z_1^{i_1} \wedge (z_0^{i_2} = z_1^{i_2} \mathbin{|}_{i_2} (z_0^{i_2} \neq z_1^{i_2} \wedge (\ldots \wedge (z_0^{i_n} = z_1^{i_n} \mathbin{|}_{i_n} (z_0^{i_n} \neq z_1^{i_n} \wedge \psi_0) \ldots);$$

$$\psi_1 := (z_0^{i_1} \neq z_1^{i_1} \mathbin{|}_{i_1} (z_0^{i_1} = z_1^{i_1} \wedge (z_0^{i_2} \neq z_1^{i_2} \mathbin{|}_{i_2} (z_0^{i_2} = z_1^{i_2} \wedge (\ldots \wedge (z_0^{i_n} \neq z_1^{i_n} \mathbin{|}_{i_n} (z_0^{i_n} = z_1^{i_n} \wedge \psi_1) \ldots).$$

The idea above is that the variables $z_0^{i_j}$, $z_1^{i_j}$ are used to encode a split of the team X_j. Using locality it is easy to see that $(\psi_0 \mathbin{|}_I \psi_1)$ and ψ are equivalent over structures of cardinality at least two. From this the claim follows in a straightforward manner. □

3.3 Data exchange in the polyteam setting

As promised, we now return to the topic of modelling data exchange in our new setting. In this section we restrict our attention to poly-atoms that are embedded dependencies. Our first goal is to define the notions of source-to-target and target poly-atoms. For this purpose we define a normal form for embedded dependencies. We call an embedded dependency $\forall\vec{x}\ \alpha(\vec{x}) \to \exists\vec{y}\ \beta(\vec{x},\vec{y})$ separated if the relation symbols that occur in α and β are distinct. A poly-atom is called separated, if the defining formula is a separated embedded dependency. In the polyteam setting this is just a technical restriction as non-separated poly-atoms can be always simulated by separated ones. Below we use the syntax $A(\vec{x}_1,\ldots,\vec{x}_l;\vec{y}_1,\ldots,\vec{y}_k)$ for separated poly-atoms. The idea is that $\vec{x}_i$s project extensions for relations used in the antecedent and $\vec{y}_j$s in the consequent of the defining formula.

Let $\mathbf{S}$ and $\mathbf{T}$ be a set of source relations and target relations from some data exchange instance, respectively. Let $\overline{X} = (S_1,\ldots,S_n,T_1,\ldots,T_m)$ be a polyteam that encodes $\mathbf{S}$ and $\mathbf{T}$ in the obvious manner. We say that an instance of a separated atom $A(\vec{x}_1,\ldots,\vec{x}_l;\vec{y}_1,\ldots,\vec{y}_k)$ is source-to-target if each $\vec{x}_i$ is a tuple of variables of the sort of S_j, for some j, and each $\vec{y}_i$ is a tuple of variables of the sort of T_j, for some j. Analogously the instance $A(\vec{x}_1,\ldots,\vec{x}_l;\vec{y}_1,\ldots,\vec{y}_k)$ is target if each $\vec{x}_i$ and $\vec{y}_j$ is a tuple of variables of the sort of T_p for some p.

Data exchange problems can now be directly studied in the polyteam setting. For example the existence-of-solution problem can be reduced to a model checking problem by using first-order quantifiers to guess a solution for the problem while the rest of the formula describes the dependences required to be fulfilled in the data exchange problem.

Example 15. A relational database schemas

$$\mathbf{S}: \quad \mathrm{P(rojects)} = \{\mathrm{name},\ \mathrm{employee},\ \mathrm{employeeposition}\};$$
$$\mathbf{T}: \quad \mathrm{E(mployees)} = \{\mathrm{name},\ \mathrm{project\_1},\ \mathrm{project\_2}\}$$

are used to store information about employees positions in different projects. We wish to check whether for a given instance of the schema $\mathbf{S}$ there exists an instance of the schema $\mathbf{T}$ that does not lose any information about for which projects employees are tasked to work and that uses the attribute name as a key. The PFO(pinc, dep)-formula

$$\alpha := \exists x_1 \exists x_2 \exists x_3\ \big(P[\{\mathrm{employee},\ \mathrm{name}\}]\ E[x_1,x_2] \mid_P P[\{\mathrm{employee},\ \mathrm{name}\}]\ E[x_1,x_3]\big) \wedge$$
$$= (x_1,(x_2,x_3))\ ;$$

when evaluated on a polyteam that encodes an instance of the schema $\mathbf{S}$, expresses that a solution for the data exchange problem exists. The variables x_1, x_2 and x_3 above are of the sort E and are used to encode attribute names name, project_1 and project_2, respectively. The dependence atom above enforces that the attribute name is a key.

4 Expressiveness

The expressiveness properties of dependence, independence, inclusion, and exclusion logic and their fragments enjoy already comprehensive classifications. Dependence logic and exclusion logic are equi-expressive and capture all downward closed ESO properties of teams [6,19]. Independence logic, whose independence atoms violate downward closure, in turn captures all ESO team properties [6]. On the other hand, the expressivity of inclusion logic has been characterised by the so-called greatest fixed point logic [7]. In this section we turn attention to polyteams and consider the expressivity of the poly-dependence logics introduced in this paper. Section 4.1 deals with logics with only uni-dependencies whereas in Section 4.2 poly-dependencies are considered.

4.1 Uni-dependencies in polyteam semantics

The following theorem displays how polyteam semantics over logics with only uni-atoms collapses to standard team semantics.

Theorem 16. Let C be a set of uni-atoms. Each formula $\alpha(\vec{x}^1,\dots,\vec{x}^n) \in \mathrm{PFO}(C)$ can be associated with a sequence of formulae $\alpha_1(\vec{x}^1),\dots,\alpha_n(\vec{x}^n) \in \mathrm{FO}(C)$ such that for all $\overline{X} = (X_1,\dots,X_n)$, where X_i is a team with domain $\vec{x}^i$,

$$M \models_{\overline{X}} \alpha(\vec{x}^1,\dots,\vec{x}^n) \;,\; \forall i = 1,\dots,n : M \models_{X_i} \alpha_i(\vec{x}^i).$$

Similarly, the statement holds vice versa.

Proof. The latter statement is clear as it suffices to set $\alpha(\vec{x}^1,\dots,\vec{x}^n) := \alpha_1(\vec{x}^1) \wedge \dots \wedge \alpha_n(\vec{x}^n)$. For the other direction, we define recursively functions f_i that map formulae $\alpha(\vec{x}^1,\dots,\vec{x}^n) \in \mathrm{PFO}(C)$ to formulae $\alpha_i(\vec{x}^i) \in \mathrm{FO}(C)$. By Proposition 14 we may assume that only disjunctions of type $\mathbin{|}_i$, for some $i \in \mathbb{N}$, may occur in α. The functions f_i are defined as follows:

- If $\beta(\vec{x}^j)$ is an atom, then $f_i(\beta) = \begin{cases} \beta & \text{if } i = j; \\ \top & \text{otherwise.} \end{cases}$
- $f_i(\beta \mathbin{|}_j \gamma) = \begin{cases} f_i(\beta) \mathbin{|}_i f_i(\gamma) & \text{if } i = j; \\ f_i(\beta) \wedge f_i(\gamma) & \text{otherwise.} \end{cases}$
- $f_i(\beta \wedge \gamma) = f_i(\beta) \wedge f_i(\gamma)$.
- For $Q \in \{\exists, \forall\}$, if $f_i(Qx^j \beta) = \begin{cases} Qx f_i(\beta) & \text{if } i = j; \\ f_i(\beta) & \text{otherwise.} \end{cases}$

We set $\alpha_i := f_i(\alpha)$ and show the claim by induction on the structure of the formula. The cases for atoms and conjunctions are trivial. We show the case for $\mathbin{|}_i$.

Let $\alpha = \beta \mathbin{|}_j \gamma$ and assume that the claim holds for β and γ. Now

$$A \models_{\overline{X}} \alpha \iff A \models_{\overline{X}[Y_j/X_j]} \beta \text{ and } A \models_{\overline{X}[Z_j/X_j]} \gamma, \text{ for some } Y_j, Z_j \subseteq X_j \text{ s.t. } Y_j \cup Z_j = X_j.$$

By the induction hypothesis, $A \models_{\overline{X}[Y_j/X_j]} \beta$ and $A \models_{\overline{X}[Z_j/X_j]} \gamma \iff A \models_{Y_j} f_j(\beta)$, $A \models_{Z_j} f_j(\gamma)$, and $A \models_{X_i} f_i(\beta); A \models_{X_i} f_i(\gamma)$ for each $i \neq j$. Thus we obtain that $A \models_{\overline{X}} \alpha$ holds if

$$A \models_{X_j} f_j(\beta) \mathbin{|}_j f_j(\gamma); \text{ and } A \models_{X_i} f_i(\beta) \text{ and } A \models_{X_i} f_i(\gamma) \text{ for each } i \neq j.$$

The above can be rewritten as

$$A \models_{X_j} f_j(\beta) \vee f_j(\gamma); \text{ and } A \models_{X_i} f_i(\beta) \wedge f_i(\gamma) \text{ for each } i \neq j.$$

The claim now follows, since $f_j(\beta) \vee f_j(\gamma) = f_j(\beta \mathbin{|}_j \gamma)$ and $f_i(\beta) \wedge f_i(\gamma) = f_i(\beta \mathbin{|}_j \gamma)$, for $i \neq j$.

The cases for the quantifiers are similar.

This theorem implies that poly-atoms which describe relations between two teams are beyond the scope of uni-logics. The following proposition illustrates this for PFO(dep).

Proposition 17. The poly-constancy atom $=(x^1/x^2)$ cannot be expressed in PFO(dep).

Proof. Assume that $=(x^1/x^2)$ can be defined by some $(x^1;x^2) \in \mathrm{PFO}(\mathrm{dep})$. By Theorem 16 there are $\mathrm{FO}(\mathrm{dep})$-formulae $_1(x^1)$ and $_2(x^2)$ such that for all $\overline{X} = (X_1;X_2)$, where X_i is a team with domain x^i, it holds that

$$M \models_{\overline{X}} =(x^1/x^2) \quad , \quad \forall i = 1;2 : M \models_{X_i} {}_i(x^i): \tag{2}$$

Define teams $X_1 := \{x^1 \mapsto 0\}$, $X_2 := \{x^2 \mapsto 0\}$, $Y_1 := \{x^1 \mapsto 1\}$, and $Y_2 := \{x^2 \mapsto 1\}$. Now clearly $M \models_{(X_1;X_2)} =(x^1/x^2)$, and $M \models_{(Y_1;Y_2)} =(x^1/x^2)$. Hence by (2), we obtain first that $M \models_{X_1} {}_i(x^1)$ and $M \models_{Y_2} {}_i(x^2)$, and then that $M \models_{(X_1;Y_2)} =(x^1/x^2)$, which is a contradiction. □

Using Theorem 16 we may now compare and characterise the expressivity of PFO(dep) and PFO(ind) in terms of existential second-order logic. To this end, let us first recall the ESO characterisations of open dependence and independence logic formulae. Note that $\mathrm{rel}(X)$ refers to a relation $\{(s(x_1);\ldots;s(x_n)) \mid s \in X\}$ where $x_1;\ldots;x_n$ is some enumeration of $\mathrm{Dom}(X)$.

Theorem 18 ([6,19]). Let $(\overline{x})$ be an independence logic (dependence logic, resp.) formula, and let R be an $|\overline{x}|$-ary relation. Then there is an (downward closed with respect to R, resp.) ESO-sentence (R) such that for all teams $X \neq \emptyset$, where $\mathrm{Dom}(X) = \overline{x}$,

$$M \models_X (\overline{x}) \quad , \quad (M; R := \mathrm{rel}(X)) \models (R)$$

The same statement holds also vice versa.

It is now easy to see that Theorems 16 and 18 together imply that PFO(dep) captures all conjunctions of downward closed ESO properties of teams whereas PFO(ind) captures all such properties.

Theorem 19. Let $(\overline{x}^1;\ldots;\overline{x}^n)$ be a PFO(ind) (PFO(dep), resp.) formula where $\overline{x}^i$ is a sequence of variables from $\mathrm{Var}(i)$. Let R_i be an $|\overline{x}^i|$-ary relation symbol for $i = 1;\ldots;n$. Then there are (downward closed with respect to R_i, resp.) ESO-sentences $_1(R_1);\ldots; {}_n(R_n)$ such that for all polyteams $\overline{X} = (X_1;\ldots;X_n)$ where $\mathrm{Dom}(X_i) = \overline{x}^i$ and $X_i \neq \emptyset$;

$$M \models_{\overline{X}} (\overline{x}^1;\ldots;\overline{x}^n) \quad , \quad (M; R_1 := \mathrm{rel}(X_1);\ldots;R_n := \mathrm{rel}(X_n)) \models {}_1(R_1) \wedge \ldots \wedge {}_n(R_n):$$

The same statement holds also vice versa.

4.2 Poly-dependencies in polyteam semantics

Next we consider poly-dependencies in polyteam semantics.

Lemma 20. The following equivalences hold:

$$= \overline{x}^1;\overline{y}^1/\overline{u}^2;\overline{v}^2 \qquad \overline{y}^1/\overline{y}^1 \;?_{\overline{x}^1;\overline{u}^2/\overline{x}^1}\; \overline{v}^2/\overline{y}^1; \tag{3}$$

$$= \overline{x}^1;y^1/\overline{u}^2;v^2 \qquad \forall z^1(y^1 = z^1 \;\_^1\; \overline{x}^1z^1 \mid \overline{u}^2v^2); \tag{4}$$

$$\overline{x}^1 \perp \overline{u}^2 \qquad \overline{x}^1/\overline{u}^2 \;?\; ;=;; \tag{5}$$

$$\overline{x}^1 \perp \overline{u}^2 \qquad \forall \overline{v}^2(\overline{x}^1 \mid \overline{v}^2 \;\_^2\; \overline{v}^2 \perp \overline{u}^2); \tag{6}$$

$$\overline{x}^1 \mid \overline{u}^2 \qquad \exists y^1z^1v^2w^2(= \overline{x}^1;y^1z^1/\overline{u}^2;v^2w^2 \wedge y^1 = z^1 \wedge v^2 \neq w^2); \tag{7}$$

$$\overline{x}^1 \mid \overline{u}^2 \qquad \exists \overline{y}^1(\overline{u}^2 \perp \overline{y}^1 \wedge \overline{x}^1 \mid \overline{y}^1); \tag{8}$$

$$\overline{y}^2/\overline{y}^1 \;?_{\overline{x}^2;\overline{x}^3/\overline{x}^1}\; \overline{z}^3/\overline{z}^1 \qquad \forall \overline{p}^2\overline{q}^2\exists u^2v^2\forall \overline{p}^3\overline{q}^3\overline{r}^3\exists u^3v^3 \;=\; \overline{p}^2\overline{q}^2;u^2v^2/\overline{p}^3\overline{q}^3;u^3v^3 \wedge \tag{9}$$

$$u^2 = v^2 \;\_^1\; (u^2 \neq v^2 \wedge \overline{x}^2\overline{y}^2 \mid \overline{p}^2\overline{q}^2) \wedge$$

$$u^3 \neq v^3 \;\_^2\; \overline{x}^3\overline{z}^3 \mid \overline{p}^3\overline{r}^3 \;\_^2\; \overline{p}^3\overline{q}^3\overline{r}^3 \perp \overline{x}^1\overline{y}^1\overline{z}^1 \;:$$

Proof. The equivalences (3){(8) are straightforward and (9) is analogous to the corresponding translation in the team semantics setting (see [6]). □

The following theorem compares the expressive powers of different polyteam-based logics. Observe that the expressivity of the logics with two poly-dependency atoms remains the same even if either one of the atoms has the standard team semantics interpretation.

Theorem 21. The following equivalences of logic hold:

(1) PFO(pdep) ≡ PFO(pexc),
(2) PFO(pind) ≡ PFO(pexc; inc) ≡ PFO(pinc; exc) ≡ PFO(pdep; inc) ≡ PFO(pinc; dep) ≡ PFO(pdep; ind) ≡ PFO(pexc; ind) ≡ PFO(pinc; ind).

Proof. Item (1) follows by equations (4) and (7). Item (2) follows from the below list of relationships:

- PFO(pind) ≡ PFO(pexc; inc) by (4), (6), and (9).
- PFO(pexc; inc) ≡ PFO(pinc; exc) by (6) and (8).
- PFO(pexc; inc) ≡ PFO(pdep; inc) by (4) and (7).
- PFO(pinc; exc) ≡ PFO(pinc; dep), since exclusion (dependence, resp.) atoms can be described in FO(dep) (FO(exc), resp.) [6].
- PFO(pdep; inc) ≡ PFO(pdep; ind), PFO(pexc; inc) ≡ PFO(pexc; ind), and PFO(pinc; dep) ≡ PFO(pinc; ind) since inclusion atoms can be described in FO(ind) [6] and dependence atoms by independence atoms [9].
- PFO(pdep; ind) ≡ PFO(ind), PFO(pexc; ind) ≡ PFO(ind), and PFO(pinc; ind) ≡ PFO(pind) by (3), (5), and (7).

□

Next we show the analogue of Theorem 18 for polyteams.

Theorem 22. Let $(R_1, \dots, R_n)$ be an ESO sentence. Then there is a PFO(pdep; inc) formula $(\bar{x}^1, \dots, \bar{x}^n)$, where $|\bar{x}^i| = ar(R_i)$, such that for all polyteams $\overline{X} = (X_1, \dots, X_n)$ with $Dom(X_i) = \bar{x}^i$ and $X_i \neq \emptyset$,

$$M \models_{\overline{X}} (\bar{x}^1, \dots, \bar{x}^n) \iff (M, R_1 := rel(X_1), \dots, R_n := rel(X_n)) \models (R_1, \dots, R_n).$$

The statement holds also vice versa.

Proof. The direction from PFO(pdep; inc) to ESO is proven by a translation similar to the one from dependence logic to ESO in [22]. We show only the opposite direction. Analogously to [6], we can rewrite $(R_1, \dots, R_n)$ as

$$\exists \bar{f} \forall \bar{u} \Big(\bigwedge_{i=1}^{n} (R_i(\bar{u}_i) \Leftrightarrow f_{2i-1}(\bar{u}_i) = f_{2i}(\bar{u}_i)) \wedge \alpha(\bar{u}, \bar{f}) \Big)$$

where $\bar{f} = f_1, \dots, f_{2n}, \dots, f_m$ is a list of function variables, α is a quantifier-free formula in which no R_i appears, each $\bar{u}_i$ is a subsequence of $\bar{u}$, and each f_i occurs only as $f_i(\bar{u}_{j_i})$ for some fixed tuple $\bar{u}_{j_i}$ of variables. For instance, $j_i = i/2$ for even $i \leq 2n$.

Let $\bar{b}^i$ be sequences of variables of sort i such that $|\bar{b}^i| = |\bar{u}_i|$, and let $\bar{u}^1 \bar{y}^1$ be a sequence of variables of sort 1 such that $\bar{u}^1$ is a copy of $\bar{u}$ and $\bar{y}^1 = y_1^1, \dots, y_m^1$. We define $(\bar{x}^1, \dots, \bar{x}^n)$ as the formula

$$\forall \bar{b}^1 \exists z_0^1 z_1^1 \dots \forall \bar{b}^n \exists z_0^n z_1^n \forall \bar{u}^1 \exists \bar{y}^1 (\alpha_0 \wedge \alpha_1 \wedge \alpha'(\bar{u}^1, \bar{y}^1))$$

where

$$\alpha_0 := \bigwedge_{i=1} = \vec{b}^i, z_0^i \wedge = \vec{b}^i, z_1^i \wedge ((\vec{b}^i \; \vec{x}^i \wedge z_0^i = z_1^i) \perp_i (\vec{x}^i \mid \vec{b}^i \wedge z_0^i \neq z_1^i));$$

$$\alpha_1 := \bigwedge_{i=1} = u_i^1, y_{2i-1}^1 / \vec{b}^i, z_0^i \wedge = u_i^1, y_{2i-1}^1 / \vec{b}^i, z_1^i \wedge \bigwedge_{i=n+1} = u_{j_i}^1, y_i^1 ;$$

and $\mathcal{A}'(\bar{u}^1, \bar{y}^1)$ is obtained from $\mathcal{A}(\bar{u}, \bar{f})$ by replacing $\bar{u}$ pointwise with $\bar{u}^1$ and each $f_i(\bar{u}_{j_i})$ with y_i^1. Above, α_0 amounts to the description of the characteristic functions f_{2i-1} and f_{2i}. We refer the reader to [6] to check that $M \models_{\bar{X}} \alpha_0$ if for all i the functions $s(\vec{b}^i) \mapsto s(z_0^i)$ and $s(\vec{b}^i) \mapsto s(z_1^i)$ determined by the assignments $s \in X_i$ agree on $s(\vec{b}^i)$ exactly when $s(\vec{b}^i) \in \mathrm{rel}(X_i)$. The poly-dependence atoms in α_1 then transfer these functions over to the first team, and the dependence atoms in α_1 describe the remaining functions. As in [6], it can now be seen that $\mathcal{A}'$ correctly simulates $\mathcal{A}$. Since exclusion atoms can be expressed in dependence logic, the claim then follows. □

By item (2) of Theorem 21 the result of Theorem 22 extends to a number of other logics as well. For instance, we obtain that poly-independence logic captures all ESO-properties of polyteams. The proof of Theorem 22 can be now easily adapted to show that poly-exclusion and poly-dependence logic capture all downward closed ESO-properties of polyteams.

Theorem 23. Let $(R_1, \ldots, R_n)$ be an ESO-sentence that is downward closed with respect to R_i. Then there is a PFO(pdep) formula $\mathcal{A}(\bar{x}^1, \ldots, \bar{x}^n)$, where $|\bar{x}^i| = \#R_i$, such that for all polyteams $\bar{X} = (X_1, \ldots, X_n)$ with $\mathrm{Dom}(X_i) = \bar{x}^i$ and $X_i \neq \emptyset$,

$$M \models_{\bar{X}} \mathcal{A}(\bar{x}^1, \ldots, \bar{x}^n) \;, \; (M, R_1 := \mathrm{rel}(X_1), \ldots, R_n := \mathrm{rel}(X_n)) \models (R_1, \ldots, R_n).$$

The statement holds also vice versa.

Proof. The direction from PFO(pdep) to ESO is again similar to the standard translation of [22]. For the other direction, let $(R_1, \ldots, R_n)$ be an ESO-sentence in which the relations R_i appear only negatively. As in the proof of Theorem 22 and by downward closure we may transform it to an equivalent form (see [19] for details)

$$\exists \bar{f} \forall \bar{u} \bigwedge_{i=1} (\neg R_i(\bar{u}_i) \vee f_{2i-1}(\bar{u}_i) = f_{2i}(\bar{u}_i)) \wedge \mathcal{A}(\bar{u}, \bar{f})$$

Now the translation $\mathcal{A}(\bar{x}^1, \ldots, \bar{x}^n)$ is defined analogously to the proof of Theorem 22 except for α_0 which is redefined as

$$\alpha_0 := \bigwedge_{i=1} = \vec{b}^i, z_0^i \wedge = \vec{b}^i, z_1^i \wedge (\vec{x}^i \mid \vec{b}^i \perp_i z_0^i = z_1^i).$$

Finally the claim follows by eliminating the exclusion atoms from α_0.

5 Conclusion

In this article we have laid the foundations of polyteam semantics in order to facilitate the fruitful exchange of ideas and results between team semantics and database theory. Our results show that many of the familiar properties and results from team semantics

carry over to the polyteam setting. In particular, we identi ed a natural polyteam analogue of dependence atoms and gave a complete axiomatisation for the associated implication problem. It is an interesting task to develop axiomatic characterisations for these new logics (cf. [20,10]). Another interesting issue is to study the expressive power of various syntactic fragments of logics over polyteams.

References

1. William W. Armstrong. Dependency Structures of Data Base Relationships. In Proc. of IFIP World Computer Congress, pages 580{583, 1974.
2. Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. J. Comput. Syst. Sci., 28(1):29{59, 1984.
3. Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. Approximation and dependence via multiteam semantics. In Proceedings of FoIKS 2016, pages 271{291, 2016.
4. Arnaud Durand, Juha Kontinen, and Heribert Vollmer. Expressivity and complexity of dependence logic. In Samson Abramsky, Juha Kontinen, Jouko Vaananen, and Heribert Vollmer, editors, Dependence Logic: Theory and Applications, pages 5{32. Springer International Publishing, Cham, 2016.
5. Ronald Fagin, Phokion G. Kolaitis, Rene J. Miller, and Lucian Popa. Data exchange: semantics and query answering. Theoretical Computer Science, 336(1):89 { 124, 2005.
6. Pietro Galliani. Inclusion and exclusion dependencies in team semantics: On some logics of imperfect information. Annals of Pure and Applied Logic, 163(1):68 { 84, 2012.
7. Pietro Galliani and Lauri Hella. Inclusion logic and xed point logic. In Proc. CSL, pages 281{295, 2013.
8. Dan Geiger, Azaria Paz, and Judea Pearl. Axioms and algorithms for inferences involving probabilistic independence. Information and Computation, 91(1):128{141, 1991.
9. Erich Gradel and Jouko A. Vaananen. Dependence and independence. Studia Logica, 101(2):399{410, 2013.
10. Miika Hannula. Axiomatizing rst-order consequences in independence logic. Ann. Pure Appl. Logic, 166(1):61{91, 2015.
11. Miika Hannula. Reasoning about embedded dependencies using inclusion dependencies. In Proceedings of LPAR-20, pages 16{30, 2015.
12. Miika Hannula and Juha Kontinen. A nite axiomatization of conditional independence and inclusion dependencies. Inf. Comput., 249:121{137, 2016.
13. Miika Hannula, Juha Kontinen, and Sebastian Link. On the nite and general implication problems of independence atoms and keys. J. Comput. Syst. Sci., 82(5):856{877, 2016.
14. Christian Herrmann. On the undecidability of implications between embedded multivalued database dependencies. Information and Computation, 122(2):221 { 235, 1995.
15. Wilfrid Hodges. Compositional Semantics for a Language of Imperfect Information. Journal of the Interest Group in Pure and Applied Logics, 5 (4):539{563, 1997.
16. Paris C. Kanellakis. Elements of relational database theory. In Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), pages 1073{1156. 1990.
17. Juha Kontinen, Antti Kuusisto, and Jonni Virtema. Decidability of Predicate Logics with Team Semantics. In Proceedings of MFCS 2016, pages 60:1{60:14, 2016.
18. Juha Kontinen, Sebastian Link, and Jouko A. Vaananen. Independence in database relations. In Proc. 20th WoLLIC, volume 8071 of LNCS, pages 179{193. Springer, 2013.
19. Juha Kontinen and Jouko Vaananen. On de nability in dependence logic. Journal of Logic, Language and Information, 3(18):317{332, 2009.
20. Juha Kontinen and Jouko A. Vaananen. Axiomatizing rst-order consequences in dependence logic. Ann. Pure Appl. Logic, 164(11):1101{1117, 2013.
21. Antti Kuusisto. A double team semantics for generalized quanti ers. Journal of Logic, Language and Information, 24(2):149{191, 2015.
22. Jouko Vaananen. Dependence Logic. Cambridge University Press, 2007.
23. Jouko Vaananen. The logic of approximate dependence. In Can Baskent, Lawrence S. Moss, and Ramaswamy Ramanujam, editors, Rohit Parikh on Logic, Language and Society, pages 227{234. Springer International Publishing, Cham, 2017.